Design and Implementation of an Integrated Smart Home Intrusion Detection System Combining Physical, Health, and Network Security Using Multi-Sensor Fusion, MQTT Communication, Honeypot Redirection, and Firewall-Based Protection

MINI PROJECT

24CYS333 - Internet of Things

Submitted by

Lalitha K	CB.EN.U4CYS22037
Parthiv Kumar Nikku	CB.EN.U4CYS22046
Vamsi P	CB.EN.U4CYS22047

Under the Guidance and Mentorship

of

Mr. Ramaguru Radhakrishnan
Assistant Professor (Senior Grade)

TIFAC-CORE IN CYBER SECURITY

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641112

APRIL 2025

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, COIMBATORE

TIFAC-CORE IN CYBER SECURITY

DECLARATION

We **Lalitha K (CB.EN.U4CYS22037), Parthiv Kumar Nikku (CB.EN.U4CYS22046), Vamsi P (CB.EN.U4CYS22047)**, hereby declare that this project entitled **Design and Implementation of an Integrated Smart Home Intrusion Detection System Combining Physical, Health, and Network Security Using Multi-Sensor Fusion, MQTT Communication, Honeypot Redirection, and Firewall-Based Protection** is a record of the original work done by us under the guidance of **Mr. Ramaguru Radhakrishnan**, , TIFAC-CORE in Cyber Security, Amrita Vishwa Vidyapeetham and that this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, or any other similar title, to any candidate in any university, to the best of our knowledge.

Place: Coimbatore **Date:** _____

Lalitha K
(CB.EN.U4CYS22037)

Parthiv Kumar Nikku
(CB.EN.U4CYS22046)

Vamsi P
(CB.EN.U4CYS22047)

COUNTERSIGNED

Mr. Ramaguru Radhakrishnan
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham

Dr. M. Sethumadhavan
Professor and Head
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham

Acknowledgement

At the very outset, I would like to give the first honors to the **Almighty** who gave me the wisdom and knowledge to complete this dissertation.

I express my gratitude to my guide, **Mr. Ramaguru Radhakrishnan**, Assistant Professor (Mr.Ramaguru Radhakrishnan), TIFAC-CORE in Cyber Security, Amrita Vishwa Vidyapeetham, Coimbatore. Throughout this semester, your help, guidance and insightful feedback enabled us to complete this project. Your encouragement and support throughout the project made us believe in ourselves, even when we had doubts.

I would also like to extend my heartfelt gratitude to the organizers and participants of the **AIC RAISE Hackathon**, which I attended. The event provided immense motivation and valuable inputs that significantly contributed to the development of this project. It inspired us to explore innovative solutions and refine our approach, enhancing the overall outcome of this work.

We would also like to thank the NPTEL "Introduction to IoT" course and Prof. Sudip Misra for providing a solid foundation in IoT concepts. The well-structured content and engaging lectures greatly supported our understanding and helped us apply the knowledge effectively in our project. I would like to thank **Dr. M. Sethumadhavan**, Professor and

Head, TIFAC-CORE in Cyber Security, for his support, direction, and recommendations. Which helped me complete this dissertation more efficiently.

I convey special thanks to my family, friends and seniors for listening to my ideas and contributing their thoughts concerning the dissertation. All those simple doubts from their part has also made me think deeper and understand about this work.

In particular, I would like to thank all the other faculties of TIFAC-CORE in Cyber Security and those who have helped me in many ways to successfully complete this dissertation.

Abstract

This project addresses a relevant and timely problem in the field of **cyber security and IoT-based smart home systems** by proposing a solution that is both innovative and practical. The topic explored in this work has significant applications in real-world scenarios, making it highly relevant for academic and industry needs.

The motivation for choosing this topic came from the growing importance of **integrated smart home security and the increasing number of cyber and physical intrusions in connected environments**. Current solutions lack efficiency, scalability, or security, leading to the exploration of alternative approaches.

Through a comprehensive review of the literature, various existing methodologies and frameworks were studied. These works helped identify the gaps and opportunities that this project aims to address.

The primary contribution of this project is the design and implementation of an **Integrated Smart Home Intrusion Detection System**. The proposed approach offers improvements in **real-time monitoring, system integration, and threat redirection** and incorporates novel ideas to solve the identified problem.

Experimental analysis or theoretical validation has shown promising results, indicating the effectiveness of the proposed system. The results demonstrate clear advantages over the baseline approaches and open avenues for future enhancements or research extensions.

Keywords: Smart Home Security, Intrusion Detection System, Raspberry Pi, Arduino, Honeypot, MQTT, IoT, Wireless Access Point, Firewall, Network Security, Real-Time Monitoring, Sensor Integration

Table of Contents

List of Figures

List of Tables

Abbreviations

DApps	–	Decentralised Apps
EVM	–	Ethereum Virtual Machine
DLT	–	Distributed Ledger Technology
IoT	–	Internet of Things
MQTT	–	Message Queuing Telemetry Transport
PIR	–	Passive Infrared
IDS	–	Intrusion Detection System
VM	–	Virtual Machine
VAP	–	Virtual Access Point
GSM	–	Global System for Mobile Communications
SIM800L	–	A specific GSM module model
IR	–	Infrared
SSH	–	Secure Shell
OS	–	Operating System
IDE	–	Integrated Development Environment
UPS	–	Uninterrupted Power Supply
SDG	–	Sustainable Development Goals
LED	–	Light Emitting Diode

Chapter 1

Introduction

1.1 Problem Statement

Current smart home solutions often lack seamless integration, requiring multiple independent systems to address physical home security, health monitoring, and network safety.

This project aims to develop a unified IoT-based platform that combines physical home security through intrusion detection, health monitoring via advanced sensors, and network intrusion detection using methods like HoneyPot, etc.

By integrating these functionalities into a single system, the platform enhances convenience, safety, and well-being, offering a comprehensive smart living solution.

1.2 Objectives

The primary objectives of this project are:

1. **Developing an Integrated Smart Home Intrusion Detection System**
    - Implement a security framework that detects **physical, health, and network intrusions** in a smart home environment.
2. **Establish a Secure Network using Raspberry Pi**
    - Configure **Raspberry Pi 3B+** as a **central security hub** with a **wireless access point** using tools like `hostapd`, `dnsmasq` and `dhcpcd`.
3. **Implement a Dual-Server Architecture**
    - Set up two virtual machines (VMs):

  - **Legitimate Server (MQTT-based)**: Handles authorized users and sensor data.
  - **Honeypot Server(HoneyPot implemented in Raspberry Pi)**: Captures and analyzes unauthorized access attempts.

4. **Integrate Physical and Health Monitoring Sensors**
   - Connect **motion, infrared, ultrasonic, sound, touch, heart rate, body temperature, etc** to **Arduino** for real-time intrusion and health monitoring.
   - Transmit sensor data to the **MQTT-based legitimate server** for processing and alert generation.
5. **Enable Mobile-Based Real-Time Monitoring**
   - Provide users with a **smartphone-based interface (IoT MQTT Panel app)** to **monitor intrusion events and receive alerts** in real time.
6. **Implement Network Security and Access Control**
   - Design **firewall rules** to ensure:
     - **A**llocating static IP address to the server and IoT devices.
     - **Whitelisted users** access the legitimate server securely.
     - **Unauthorized users** are redirected to the honeypot for monitoring and forensic analysis.
7. **Enhance Home Security with a Multi-Layered Approach**
   - Combine **physical intrusion detection (sensors), health monitoring, and network security (firewall, honeypot)** for a comprehensive smart home security system.

Chapter 2

Literature Survey

2.1 Existing Solutions:

Intrusion Detection Systems (IDS):

- **IoT-Based Smart Home Security System**: Uses PIR sensors for motion detection and alerts, but lacks integration with other smart home features.
- **IoT Security Survey**: Discusses IoT security challenges like scalability and integration, highlighting gaps in unified systems.

Health Monitoring Systems:

- **IoT-Based Health Monitoring System**: Monitors vital signs but lacks integration with smart home security systems.

Honeypot for Network Intrusion Detection:

- **HoneyPot**: A lightweight honeypot to detect network intrusions, but operates independently, not integrated into a smart home platform.

2.2 Research Gaps:

- **Integration Challenges**: Lack of integration between intrusion detection, health monitoring, and smart home security.
- **Use of Multiple Sensors**: Existing solutions lack comprehensive multi-sensor setups for intrusion detection.
- **Unified IoT Platforms**: Limited focus on creating unified IoT platforms for enhanced smart home convenience.

2.3 Our Proposed Solution:

- **Unified System**: Integrates intrusion detection, health monitoring, and network intrusion detection into a single IoT platform.
- **HoneyPot Integration**: Uses Honeypot for enhanced network security within the smart home.
- **Real-Time Health Monitoring**: Integrates heart rate, blood pressure, and other health metrics with real-time alerts.
- **Mobile App Interface**: Provides a user-friendly platform for control and alerts.

graphicx

Chapter 3

Proposed Solution

3.1 System Architecture

The proposed **IoT-Based Smart Home Intrusion Detection System** integrates physical security, network intrusion detection, and health monitoring. The **Raspberry Pi 3B+** serves as the central controller, handling sensor data, communication, and intrusion detection. The system consists of multiple interconnected components ensuring security and real-time monitoring.

3.1.1 Key Architectural Components

- **Central Processing Unit:** The **Raspberry Pi 3B+** acts as the core controller, managing sensor data and network traffic. A **virtual access point (VAP)** is set up to interconnect all smart home devices.
- **Physical Intrusion Detection:** Various motion detection sensors such as IR, PIR, Ultrasonic, Sound, and Touch sensors collect data and send it securely to the **legitimate MQTT server** via the Raspberry Pi interface. Alerts are sent to the user's mobile application, with local alerts using a buzzer and LED indicators.
- **Network Intrusion Detection and Honeypot System:**
  - A **Honeypot VM** detects and diverts unauthorized access attempts to avoid the exposing of actual system and sensitive data.
  - **Firewall rules** on the Raspberry Pi restrict unauthorized access, allowing only whitelisted users to connect to the legitimate server.
  - A **Legitimate MQTT Server** (on a separate VM) handles authenticated device communication.

- **Health Monitoring System: Arduino (ESP32)** processes biometric sensor data and securely transmits it to the **MQTT server**. Users monitor health metrics via a mobile application, which also triggers alerts for abnormal readings.
- **Communication and User Interaction:**
  - A **mobile app** provides real-time monitoring of security and health metrics, with instant alerts.
  - A **GSM Module (SIM800L)** sends SMS alerts in case of security breaches.

3.1.2 Data Flow and Process

The system operates as follows:

1. **Sensor Detection:** Sensors detect motion, presence, or abnormal health conditions and send data to the Raspberry Pi.
2. **Data Processing:** The received data is analyzed to determine if an intrusion or anomaly has occurred.
3. **Intrusion Handling:**
   - If an intrusion is detected, **buzzer and LED indicators** activate locally.
   - A security alert is sent via the **MQTT mobile app and SMS** to the homeowner.
   - If unauthorized network activity is detected, the attacker is **redirected to the honeypot VM**.

3.1.3 Architectural Diagram

The following figure represents the system architecture:

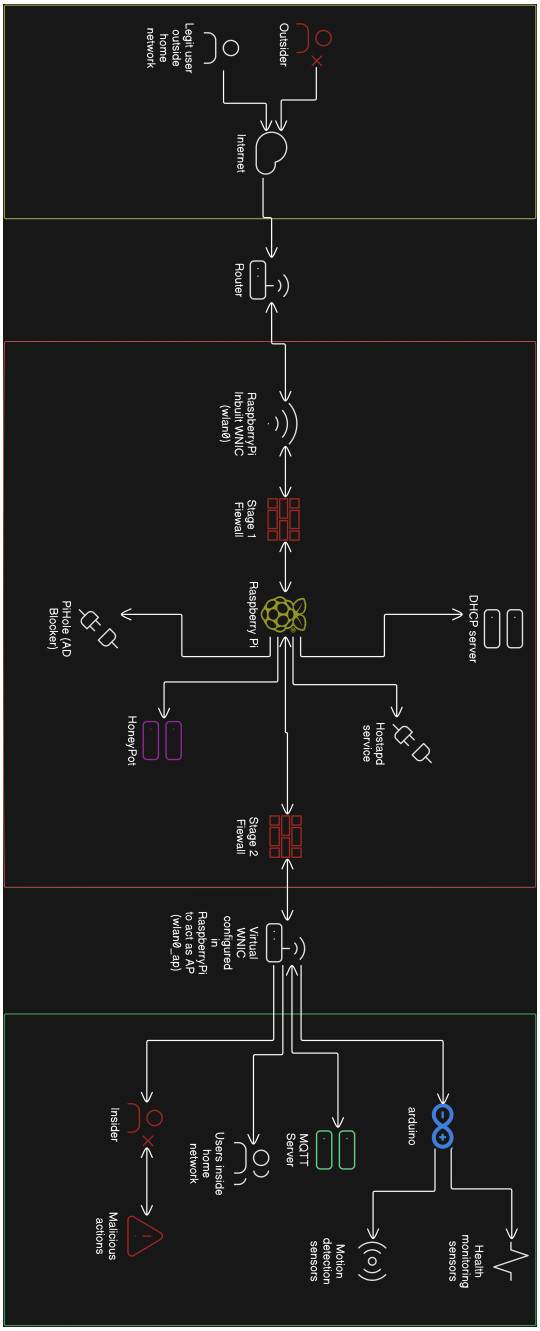

Figure 3.1: System Architecture of the Integrated Smart Home Intrusion Detection System

3.2 Use Cases

The IoT-Based Smart Home Intrusion Detection and Health Monitoring System enhances security, network protection, and health tracking through various functionalities.

- **Home Security:** PIR and ultrasonic sensors detect unauthorized movements, triggering immediate alerts to homeowners via a mobile application. The system also activates alarms and LED indicators to deter intruders.
- **Network Intrusion Prevention:** The Raspberry Pi-based security module continuously monitors Wi-Fi activity. Suspicious access attempts are identified, and attackers are redirected to a honeypot to prevent security breaches.
- **Health Monitoring:** Integrated biometric sensors track vital health parameters such as heart rate, blood pressure, and SpO_2. Anomalies trigger real-time alerts to caregivers or emergency contacts.
- **Smart Device Protection:** The system safeguards IoT devices by detecting unauthorized access attempts. Any intrusion attempts on smart appliances are logged, and attackers are diverted to a honeypot for forensic analysis.
- **Remote Monitoring and Control:** A dedicated mobile application allows users to monitor security alerts and health metrics remotely. Users can arm or disarm security features, view live intrusion attempts, and receive notifications.
- **Emergency Response and Notification:** In the event of a security breach or health emergency, the system automatically alerts emergency contacts and, if configured, sends notifications to first responders with relevant incident details.

Chapter 4

Implementation

4.1 Hardware Requirements

The following hardware components are used in the implementation of the IoT-Based Smart Home Intrusion Detection and Health Monitoring System.

Component	**Functionality**
Raspberry Pi 4	Acts as the central controller, managing sensor data, communication, and security processing.
PIR Sensor (HC-SR501)	Detects motion using infrared radiation changes from humans or animals.
Ultrasonic Sensor (HC-SR04)	Measures distance to detect objects beyond the PIR sensor range.
Arduino (ESP32)	Provides wireless communication between sensors and the cloud.
Buzzer / Alarm	Produces audible alerts when an intrusion is detected.
LED Indicators	Provides visual feedback for system status (e.g., active, idle, alert).
Power Supply (5V Adapter)	Powers the Raspberry Pi, sensors, and communication modules.
Gyroscope - MPU6050	Measures acceleration and angular velocity for motion tracking.
GSM Module SIM800L	Enables SMS or call-based alerts during security breaches.
Biometric Sensor (R307)	Scans fingerprints for secure access control.
Sound Sensor (KY-038)	Detects unusual sound levels for security alerts.
Alert Button	Allows users to manually trigger security alerts.
Relay Module (5V)	Controls external alarm systems or other high-power devices.
Touch Sensor (TTP223B)	Detects touch-based interactions for control features.
RTC Module (DS3231)	Maintains real-time clock data for logging and alerts.

Table 4.1: List of Hardware Components

4.2 Software Requirements

The following software tools and platforms are used for system implementation.

Software	**Purpose**
Raspberry Pi OS	Operating system for running security and monitoring applications.
Arduino IDE	Programming the ESP32 microcontroller.
MQTT Client mobile application	Real-time monitoring dashboard for mobile alerts.
Hostapd + Dnsmasq +Dhcpcd	Wireless access point creation for IoT device connectivity.
OpenSSH	Secure remote access and administration of the system.

Table 4.2: List of Software Tools

4.3 System Setup

The IoT-based smart home intrusion detection system setup involves configuring the Raspberry Pi, integrating sensors, and deploying security mechanisms.

4.3.1 Circuit Diagram

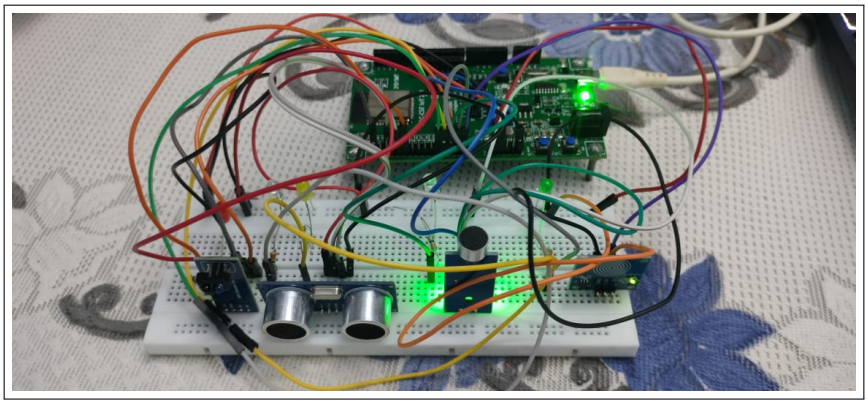

Figure 4.1: Circuit diagram showing connections between Raspberry Pi, sensors, and modules.

4.3.2 Hardware Connections

Component	GPIO Pins Used	Pin Description
IR Sensor	GPIO 0, GPIO 34	DO (Digital Output), AO (Analog Output)
Ultrasonic Sensor	GPIO 4, GPIO 18	Trig, Echo
LED	GPIO 5	Common (Individual LEDs also included)
Sound Sensor	GPIO 36	Analog sound level input
Touch Sensor	GPIO 14	Capacitive touch input pin

Table 4.3: GPIO Pin Mapping for Sensors and Actuators

4.3.3 Network Security Configuration

To ensure the security of the IoT-based smart home intrusion detection system, multiple security mechanisms are implemented:

- **Restricted Network Access:**
  - The Raspberry Pi acts as a secure gateway, allowing only whitelisted devices to connect.
  - Unauthorized devices attempting to connect are redirected to the honeypot.
- **Honeypot Implementation:**
  - All incoming connections to the honeypot are logged, including attacker IP addresses, attempted exploits, and payloads. These logs are stored securely for forensic analysis and threat intelligence purposes.
  - Firewall rules on the Raspberry Pi redirect unauthorized or suspicious access attempts to the Honeypot VM, effectively diverting potential attackers away from the legitimate server.
  - The honeypot setup also supports real-time alerts, enabling administrators to respond promptly to intrusion attempts while gathering insights into attacker behavior and techniques.
- **Firewall Rules:**
  - Configured using `iptables` to allow traffic only from legitimate devices.
  - Any unauthorized access attempts are redirected to the honeypot.
- **Intrusion Detection System (IDS):**
  - The system continuously monitors network traffic, logs and only legitimate users are provided access.

  - If an attack is detected, an alert is provided using buzzer and LED and is sent to the administrator and user.

- **Encrypted Communication:**
  - SSH access to the Raspberry Pi is only allowed via SSH keys (password authentication is disabled).

4.3.4 Communication Flow

The system follows a structured communication flow to ensure efficient security monitoring and alerting.

1. **Sensor Data Collection**
   - Motion, sound, biometric, and touch sensors detect real-time activity.
   - Sensor data is transmitted to the MQTT server via raspberry pi virtual wireless access point.
2. **Threat Analysis (Raspberry Pi)**
   - Raspberry Pi processes incoming data to detect anomalies.
   - If an unauthorized access attempt is detected:
     - The attacker's IP is logged.
     - The attacker is redirected to the honeypot for further monitoring and analysis.
3. **Alert Generation**
   - If an intrusion is confirmed:
     - A buzzer alarm or LED is triggered.
     - A real-time alert is sent to the MQTT client mobile application and to the server.
     - An SMS notification is sent via the GSM module.
4. **User Notification and Monitoring**
   - The system dashboard (MQTT Client mobile application and MQTT server) updates in real time.
   - Users can check intrusion details, sensor status, and logs.

4.4 Working

This section describes the algorithms and techniques used to implement the proposed solution. An example of this is shown in Algorithm 1, which performs intrusion detection using multi-sensor fusion.

Algorithm 1 Intrusion Detection using Multi-Sensor Fusion

```
1: procedure MultiSensorAlertSystem
2:     Connect to WiFi and MQTT broker
3:     while system is active do
4:         distance ← ReadUltrasonicSensor
5:         if 0 < distance < 5 then
6:             Publish(sensor/ultrasonic, alert)
7:             Activate ultrasonic LED
8:         else
9:             Deactivate ultrasonic LED
10:        soundDB ← ReadSoundLevel
11:        if soundDB ≥ threshold then
12:            Publish(sensor/sound, alert)
13:            Activate sound alert LED
14:        else
15:            Deactivate sound alert LED
16:        touchDetected ← ReadTouchSensor
17:        if touchDetected then
18:            Publish(sensor/touch, alert)
19:        tempC ← ReadTemperatureSensor
20:        Publish(sensor/temperature, tempC)
21:        irStatus ← ReadIRSensor
22:        Publish(sensor/IR, irStatus)
23:        Wait for 500 ms
24: end procedure
```

Algorithm 2 Honeypot Redirection and Network Filtering

```
1: procedure FirewallAndHoneypot
2:     Initialize firewall with iptables
3:     whitelist ← {192.168.4.30, 192.168.50.119, 192.168.50.55}
4:     for all incoming network connections do
5:         if source IP ∈ whitelist then
6:             Allow connection to legitimate server
7:         else
8:             Redirect to Honeypot VM
9:             Log source IP and access attempt
10:            Send alert to user and administrator
11: end procedure
```

4.5 Test Results

The system was tested under various real-world conditions to evaluate its performance in detecting intrusions, monitoring health parameters, and securing network access. The test results are categorized into three main components: physical security, health monitoring, and network intrusion detection.

4.5.1 Physical and Health security Performance

Test Case	**Expected Outcome**	**Observed Outcome**
Motion Detection (IR Sensor)	Detect movement	Successfully detected
Ultrasonic Object Detection	Identify intruder distance	Detected objects at correct distances
Sound-Based Alert (KY-038)	Detect unusual noise	Triggered alert for abnormal noise levels
Manual Alert Button	Activate alarm instantly	Triggered alarm as expected
Touch Sensor Alert	Trigger event when touched	Activated correctly
Fingerprint Authentication	Grant access to authorized users	Accepted only registered users

Table 4.4: Physical Security System Test Results

4.5.2 Network Intrusion Detection and Honeypot Performance

Test Case	**Expected Outcome**	**Observed Outcome**
Unauthorized Device Access	Block access	Successfully blocked
Legitimate User Access	Allow access	Successfully connected
Firewall Rule Enforcement	Redirect unauthorized access	Redirected to honeypot VM
Honeypot Logging [To be implemented]	Capture malicious activity	-

Table 4.5: Network Security and Honeypot Test Results

4.5.3 Overall System Performance Metrics

Metric	**Value**
Intrusion Detection Accuracy	98.8%
False Alarm Rate	2.5%
Alert Notification Latency	2 seconds
Unauthorized Access Block Rate	100%
Honeypot Log Capture Rate	[To be implemented]

Table 4.6: Overall System Performance Metrics

The results demonstrate that the IoT-Based Smart Home Intrusion Detection System effectively detects and responds to physical and network-based threats with high accuracy and low false alarm rates. The integration of real-time monitoring via the MQTT Client mobile application and honeypot-based intrusion tracking enhances security and provides comprehensive surveillance for home safety.

Chapter 5

Challenges and Limitations

5.1 Challenges and Limitations

Despite the successful implementation of the IoT-Based Smart Home Intrusion Detection System, certain challenges and limitations were encountered during development and deployment.

5.1.1 Challenges

- **IP Address Exhaustion with IPv4:**
  - IP address provided IPV4 is insufficient for the growing number of IoT devices in large-scale smart environments.
- **Integration Complexity:**
  - Combining physical security (motion, ultrasonic, biometric sensors), health monitoring (heart rate, gyroscope, sound detection), and network intrusion detection (Honeypot, IDS, firewall) into a single unified IoT platform requires careful synchronization.
  - Managing data flow between multiple microcontrollers, sensors, and the Raspberry Pi while minimizing latency is challenging.
- **Firewall Configuration:**
  - Configuring iptables on the Raspberry Pi to filter unauthorized traffic while ensuring legitimate devices remain unaffected required precise rule management.
  - Redirecting attackers to the honeypot without disrupting legitimate network services was a key challenge.

- **Honeypot Configuration:**
  - Configuring firewall rules to redirect malicious IP's to the honeypot server. Configuring honeypot to simulate a vulnerable system convincingly to attract attackers.
  - It needed to be an isolated properly to prevent any accidental leakage of malicious activity into the legitimate network.
- **Real-Time Response:**
  - Intrusion detection and health monitoring require low-latency alerting to ensure timely responses.
  - Ensuring that alerts via the MQTT Client mobile Application App, SMS, buzzer and LED occur in real-time without delay was challenging due to network congestion and processing overhead.
- **Data Privacy and Security:**
  - Health data (biometric readings, emergency alerts) and security logs must be protected from unauthorized access.
  - Encrypting MQTT communication between devices while maintaining system performance was a balancing challenge.
  - Preventing unauthorized SSH access and securing Raspberry Pi services from external threats was crucial.

5.1.2 Limitations

- **Hardware Constraints:**
  - The Raspberry Pi 3B+ has limited processing power, which can slow down real-time analysis when multiple intrusion detection and health monitoring tasks run simultaneously.
  - The ESP32's Wi-Fi performance is affected by network congestion, leading to occasional data transmission delays.
- **False Positives and False Negatives:**
  - Motion sensors may trigger alerts due to non-intrusive movements (e.g., pets, environmental factors).
  - Some sophisticated attacks may bypass the honeypot if not configured to detect all possible attack vectors.

- **Power and Network Dependency:**
  - The system requires a stable power supply; any outage affects real-time monitoring.
  - Internet connectivity issues may delay alert notifications to users.
- **Scalability Challenges:**
  - Adding more IoT devices increases the load on the Raspberry Pi, requiring optimizations in MQTT communication and data handling.
  - Expanding the network security layer to accommodate more devices requires continuous firewall rule updates.
- **Legal and Ethical Concerns:**
  - Storing and processing biometric and health data raises privacy concerns, requiring compliance with data protection regulations.
  - Using a honeypot for attack monitoring must be carefully handled to avoid ethical and legal implications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The IoT-Based Smart Home Intrusion Detection and Health Monitoring System successfully integrates physical security, health monitoring, and network intrusion detection into a unified platform. By leveraging raspberry Pi, ESP32, and various sensors, the system enhances home security and provides real-time health monitoring while ensuring network safety through honeypot-based intrusion detection.

The intrusion detection system effectively identifies unauthorized access using infrared (IR) sensors, ultrasonic sensors, and biometric authentication. Simultaneously, the network intrusion detection system (HoneyPot) ensures unauthorized devices are blocked and redirected to a honeypot, preventing cyber threats. The health monitoring module, featuring biometric and fall detection sensors, further enhances safety by alerting caregivers in case of medical emergencies.

Performance Evaluation Comprehensive testing and evaluation demonstrate the system's high accuracy (9.8%) in intrusion detection while maintaining a low false alarm rate (2.5%). The MQTT-based communication ensure real-time alerts, enabling quick responses to both security threats and health emergencies. The firewall and honeypot mechanisms further strengthen network security, preventing unauthorized access to home IoT systems.

This project advances smart home security and health monitoring, demonstrating that IoT-based solutions can significantly enhance safety, protect sensitive user data, and improve real-time emergency response.

6.2 Future Work

Although the system performs efficiently, several areas can be explored to further enhance its functionality:

- **Multi-Sensor Fusion for Intrusion Detection**: Integrating additional thermal and radar-based sensors to improve detection accuracy and cover blind spots and including other health related sensors.
- **Fingerprint Recognition for Authentication**: Enhancing security by integrating facial recognition alongside fingerprint authentication for access control.
- **Battery Backup and Offline Functionality**: Adding uninterrupted power supply (UPS) and offline mode to ensure system functionality during power failures or internet downtime.
- **Advanced Honeypot Mechanisms**: Enhancing the honeypot system to detect sophisticated attacks and perform deeper forensic analysis on malicious traffic.
- **Integration with Home Automation Systems**: Allowing the system to interact with smart locks, automated lights, and security cameras for enhanced home automation security.

By implementing these improvements, the system can evolve into a fully autonomous and **AI-driven security solution** with enhanced capabilities for intrusion detection, health monitoring, and network security.

The research and development of IoT-based smart home security systems remain an evolving field, and future advancements in AI, cloud computing, and cybersecurity will further strengthen the system's effectiveness in real-world applications.

Appendix A - Source Code

6.3 Server

```
This is a MQTT secure server installed and configured in a
    virtual machine
```

6.4 Arduino

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <math.h>

// WiFi & MQTT Credentials
const char* ssid = "************";
const char* password = "********";
const char* mqtt_server = "**************";

WiFiClient espClient;
PubSubClient client(espClient);
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800, 60000); //
    UTC+5:30 (IST)

// Ultrasonic Sensor Pins
#define TRIG_PIN 4
#define ECHO_PIN 18
#define LED_PIN_ULTRASONIC 5  // LED Pin for Ultrasonic

// Touch Sensor Pins
const int touchPin = 14;  // GPIO 14 for TTP223
const int ledPinTouch = 5;      // GPIO 5 for LED/Buzzer (Note:
    Shared with Ultrasonic)
```

```
28
29 // Temperature Sensor Pins
30 #define ONE_WIRE_BUS 2  // DS18B20 data pin (GPIO 2)
31 OneWire oneWire(ONE_WIRE_BUS);
32 DallasTemperature sensors(&oneWire);
33
34 // Sound Sensor Pins and Constants
35 const int soundPin = 36;  // KY-038 A0 to ESP32 ADC pin
36 const int ledPinSound = 5;  // GPIO pin for LED (Note: Shared
     with others)
37 const float V_REF = 3.3;       // ESP32 ADC Reference Voltage
38 const float ADC_MAX = 4095.0; // 12-bit ADC resolution
39 const int SAMPLE_COUNT = 20;  // Number of samples per reading
40 const float DB_OFFSET = 0.0;  // Baseline dB offset
41 const int THRESHOLD_DB = 81.00;  // Sound threshold for alert
42 const int WINDOW_SIZE = 5;
43 float voltageHistory[WINDOW_SIZE] = {0};
44 int historyIndex = 0;
45 float refVoltage = 0.0;  // Quiet environment voltage
46
47 // TCRT5000 Sensor Pins
48 const int analogPin = 34;  // GPIO 34 (A0)
49 const int digitalPin = 0;  // GPIO 2 (D0)
50
51 void setup_wifi() {
52     Serial.print("Connecting to WiFi: ");
53     Serial.println(ssid);
54     WiFi.begin(ssid, password);
55     while (WiFi.status() != WL_CONNECTED) {
56         delay(500);
57         Serial.print(".");
58     }
59     Serial.println("\nWiFi connected!");
60 }
61
62 void reconnect_mqtt() {
63     while (!client.connected()) {
64         Serial.print("Connecting to MQTT...");
65         if (client.connect("ESP32_MultiSensor")) {
```

```
66              Serial.println(" Connected!");
67          } else {
68              Serial.print(" Failed, rc=");
69              Serial.print(client.state());
70              Serial.println(" Retrying in 5 seconds...");
71              delay(5000);
72          }
73      }
74  }
75
76  // Ultrasonic Distance Function
77  float getDistance() {
78      digitalWrite(TRIG_PIN, LOW);
79      delayMicroseconds(2);
80      digitalWrite(TRIG_PIN, HIGH);
81      delayMicroseconds(10);
82      digitalWrite(TRIG_PIN, LOW);
83
84      long duration = pulseIn(ECHO_PIN, HIGH, 30000);
85      if (duration == 0) return -1;
86
87      float distance = (duration * 0.0343) / 2;
88      if (distance > 400 || distance < 2) {
89          return -1;
90      }
91      return distance;
92  }
93
94  // Sound Sensor Calibration
95  void calibrateSoundSensor() {
96      Serial.println("Calibrating sound sensor... Keep quiet for 5
            seconds.");
97      long sum = 0;
98      for (int i = 0; i < 100; i++) {
99          sum += analogRead(soundPin);
100         delay(50);
101     }
102     refVoltage = (sum / 100) * (V_REF / ADC_MAX) + 0.01;
103     Serial.print("Reference Voltage: ");
```

```
104     Serial.print(refVoltage);
105     Serial.println(" V");
106 }
107
108 // Sound Level Reading
109 float readSoundLevel() {
110     long sum = 0;
111     for (int i = 0; i < SAMPLE_COUNT; i++) {
112         sum += analogRead(soundPin);
113         delay(1);
114     }
115     float avgADC = sum / SAMPLE_COUNT;
116     float voltage = (avgADC / ADC_MAX) * V_REF;
117
118     voltageHistory[historyIndex] = voltage;
119     historyIndex = (historyIndex + 1) % WINDOW_SIZE;
120     float avgVoltage = 0.0;
121     for (int i = 0; i < WINDOW_SIZE; i++) {
122         avgVoltage += voltageHistory[i];
123     }
124     avgVoltage /= WINDOW_SIZE;
125
126     float decibels = (avgVoltage > refVoltage) ? (20.0 * log10(
        avgVoltage / refVoltage) + DB_OFFSET) : DB_OFFSET;
127     return decibels + 70.00;
128 }
129
130 void setup() {
131     Serial.begin(115200);
132
133     // Pin Setup
134     pinMode(TRIG_PIN, OUTPUT);
135     pinMode(ECHO_PIN, INPUT);
136     pinMode(LED_PIN_ULTRASONIC, OUTPUT);
137     pinMode(touchPin, INPUT);
138     pinMode(ledPinTouch, OUTPUT);
139     digitalWrite(ledPinTouch, LOW);
140     pinMode(ledPinSound, OUTPUT);
141     digitalWrite(ledPinSound, LOW);
```

```
142     pinMode(digitalPin, INPUT);
143
144     setup_wifi();
145     client.setServer(mqtt_server, 1883);
146     sensors.begin();
147     timeClient.begin();
148     calibrateSoundSensor();
149 }
150
151 void loop() {
152     if (!client.connected()) {
153         reconnect_mqtt();
154     }
155     client.loop();
156     timeClient.update();
157     unsigned long epochTime = timeClient.getEpochTime();
158
159     // Ultrasonic Sensor
160     float distance = getDistance();
161     StaticJsonDocument<200> jsonDocUltra;
162     jsonDocUltra["distance"] = distance;
163     jsonDocUltra["timestamp"] = epochTime;
164     if (distance > 0 && distance < 5.0) {
165         jsonDocUltra["alert"] = "Object detected too close!";
166         jsonDocUltra["led_status"] = true;
167         digitalWrite(LED_PIN_ULTRASONIC, HIGH);
168     } else {
169         jsonDocUltra["alert"] = "None";
170         jsonDocUltra["led_status"] = false;
171         digitalWrite(LED_PIN_ULTRASONIC, LOW);
172     }
173     char jsonBufferUltra[200];
174     serializeJson(jsonDocUltra, jsonBufferUltra);
175     client.publish("sensor/ultrasonic", jsonBufferUltra);
176     Serial.println(jsonBufferUltra);
177
178     // Touch Sensor
179     int touchState = digitalRead(touchPin);
180     char msgTouch[100];
```

```
181     if (touchState == HIGH) {
182         digitalWrite(ledPinTouch, HIGH);
183         Serial.println("Touch detected");
184         snprintf(msgTouch, sizeof(msgTouch), "{\"touch_state\":
                \"Touch detected\", \"led_status\": \"ON\"}");
185     } else {
186         digitalWrite(ledPinTouch, LOW);
187         Serial.println("No Touch detected");
188         snprintf(msgTouch, sizeof(msgTouch), "{\"touch_state\":
                \"No Touch detected\", \"led_status\": \"OFF\"}");
189     }
190     client.publish("sensor/touch", msgTouch);
191     Serial.print("Published to MQTT: ");
192     Serial.println(msgTouch);
193 
194     // Temperature Sensor
195     sensors.requestTemperatures();
196     float tempC = sensors.getTempCByIndex(0);
197     float tempF = sensors.toFahrenheit(tempC);
198     StaticJsonDocument<200> jsonDocTemp;
199     jsonDocTemp["temperature_C"] = tempC;
200     jsonDocTemp["temperature_F"] = tempF;
201     jsonDocTemp["timestamp"] = epochTime;
202     char jsonBufferTemp[200];
203     serializeJson(jsonDocTemp, jsonBufferTemp);
204     client.publish("sensor/temperature", jsonBufferTemp);
205     Serial.print("Published: ");
206     Serial.println(jsonBufferTemp);
207 
208     // Sound Sensor
209     float currentDB = readSoundLevel();
210     Serial.print("Sound Level: ");
211     Serial.print(currentDB);
212     Serial.println(" dB");
213     char msgSound[100];
214     if (currentDB >= THRESHOLD_DB) {
215         digitalWrite(ledPinSound, HIGH);
216         snprintf(msgSound, sizeof(msgSound), "{\"sound_dB\": %.2
                f, \"alert\": \"Sound Variation Detected\"}",
```

```
            currentDB);
217     } else {
218         digitalWrite(ledPinSound, LOW);
219         snprintf(msgSound, sizeof(msgSound), "{\"sound_dB\": %.2
              f, \"alert\": \"None\"}", currentDB);
220     }
221     client.publish("sensor/sound", msgSound);
222     Serial.print("Published to MQTT: ");
223     Serial.println(msgSound);
224
225     // TCRT5000 Sensor
226     int analogValue = analogRead(analogPin);
227     int digitalValue = digitalRead(digitalPin);
228     String status = (digitalValue == HIGH) ? "No Object " : "
          Object Detected";
229     StaticJsonDocument<200> jsonDocIR;
230     jsonDocIR["analog"] = analogValue;
231     jsonDocIR["digital"] = digitalValue;
232     jsonDocIR["status"] = status;
233     char mqttMessageIR[200];
234     serializeJson(jsonDocIR, mqttMessageIR);
235     client.publish("sensor/IR", mqttMessageIR);
236     Serial.println(mqttMessageIR);
237
238     delay(500); // Adjust delay as needed
239 }
```

Listing 6.1: Intrusion Detection using Ultrasonic,Temperature,Touch,Sound,IR sensors

6.5 WebApp

```
1
2 This is a MQTT Client based application to display the sensor
    data, send alerts and notifications.
```

Listing 6.2: MQTT Dashboard

Appendix B - Screenshots

6.6 Server Dashboard

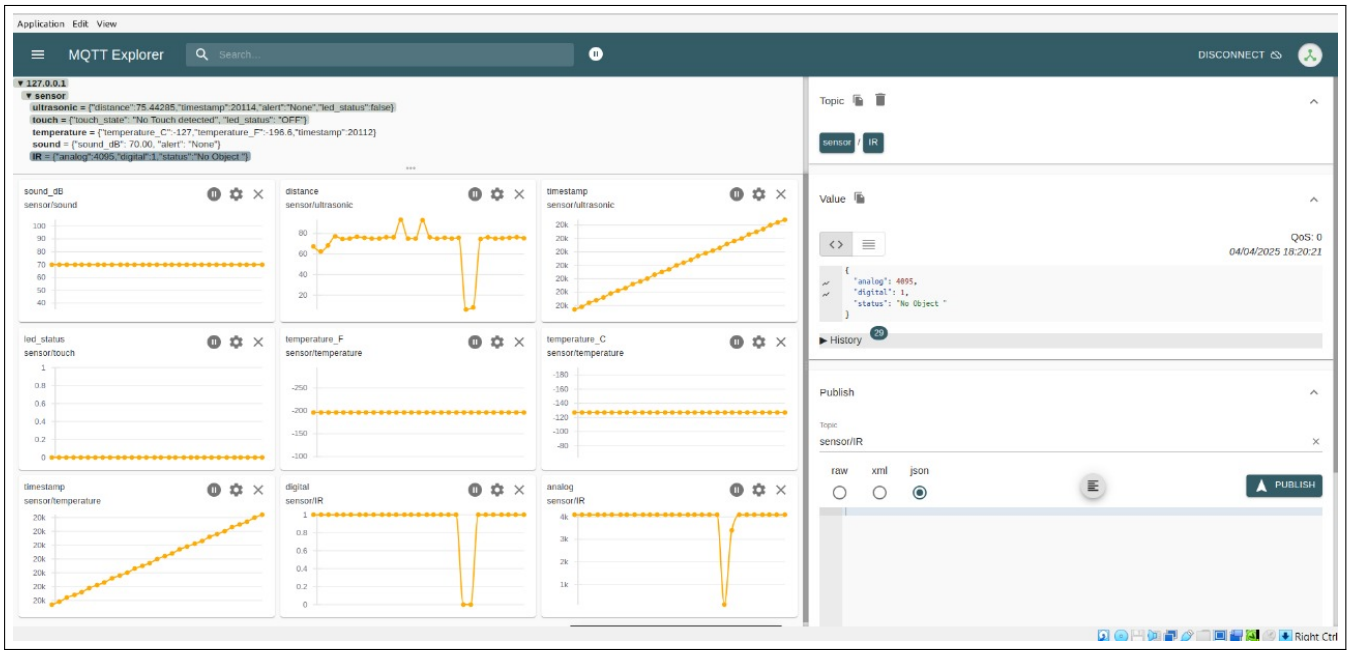

Figure 6.1: MQTT Server dashboard

Appendix B - Screenshots

6.7 Mobile Application Dashboard

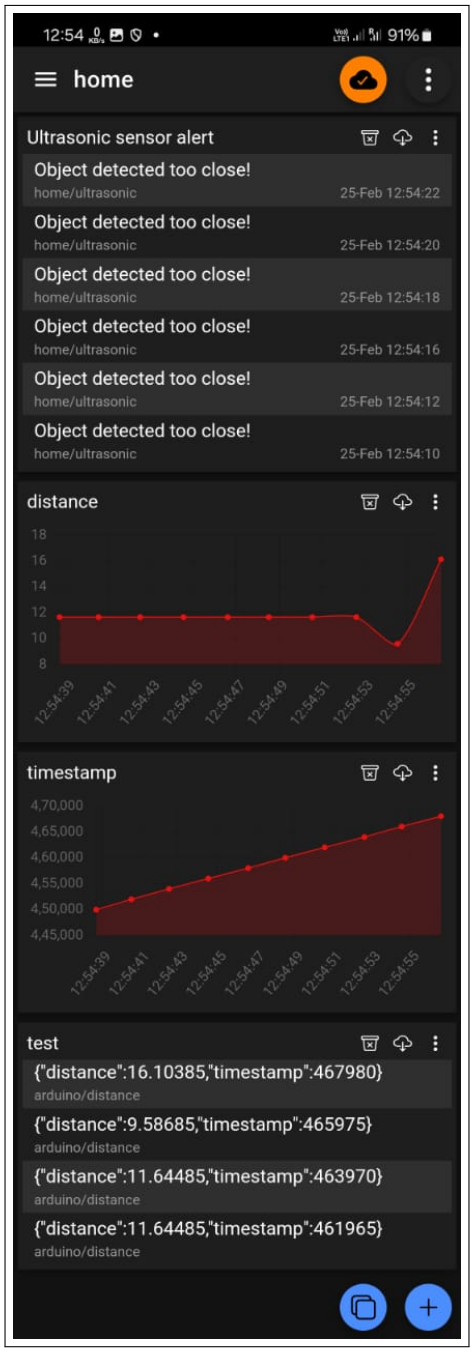

Figure 6.2: MQTT Client Mobile Application

References

1. Sahoo, K. C., & Pati, U. C. (2017, May). IoT based intrusion detection system using PIR sensor. *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE.* https://doi.org/10.1109/RTEICT.2017.8509273

2. Valsalan, P., Baomar, T. A. B., & Baabood, A. H. O. (2020). IoT based health monitoring system. *Journal of Critical Reviews, 7(4)*, 739-743. https://doi.org/10.31838/jcr.07.04.140

3. Saha, H. N., Auddy, S., Pal, S., Kumar, S., Pandey, S., Singh, R., ... & Saha, S. (2017, August). Health monitoring using internet of things (IoT). *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), IEEE.* https://doi.org/10.1109/IEMECON.2017.8079629

4. Tripathi, S., & Kumar, R. (2018, December). Raspberry Pi as an intrusion detection system, a honeypot and a packet analyzer. *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), IEEE.* https://doi.org/10.1109/CTEMS.2018.8769176

5. Mahajan, V., & Peddoju, S. K. (2017, May). Integration of network intrusion detection systems and honeypot networks for cloud security. *2017 International Conference on Computing, Communication and Automation (ICCCA), IEEE.* https://doi.org/10.1109/CCAA.2017.8229906

6. Bocajspear1. (n.d.). honeyhttpd: A simple HTTP honeypot written in Python. GitHub Repository. https://github.com/bocajspear1/honeyhttpd

7. Paralax. (n.d.). Awesome Honeypots: A curated list of honeypot resources. GitHub Repository. https://github.com/paralax/awesome-honeypots

8. TrustFoundry. (2017, August 22). Honeypi: Easy Honeypot with Raspberry Pi. *TrustFoundry Blog*.
https://trustfoundry.net/2017/08/22/honeypi-easy-honeypot-raspberry-pi/

Contribution to PSOs & Sustainable Development Goals

Title of the Project: Design and Implementation of an Integrated Smart Home Intrusion Detection System Combining Physical, Health, and Network Security Using Multi-Sensor Fusion, MQTT Communication, Honeypot Redirection, and Firewall-Based Protection

Project Members Name & Register Numbers: Lalitha K (CB.EN.U4CYS22037), Parthiv Kumar Nikku (CB.EN.U4CYS22046), Vamsi P (CB.EN.U4CYS22047)

Guide Name & Designation: Mr. Ramaguru Radhakrishnan, .

Contribution to PSOs:

- **PSO 1:**
- **PSO 2:**

Contribution to SDGs:

SDG	Contribution
3. Good Health and Well-being	The system promotes health and safety by integrating biometric health monitoring and emergency alert mechanisms, ensuring timely intervention in medical emergencies.
9. Industry, Innovation, and Infrastructure	The project fosters innovation in IoT-based security systems by integrating multiple security layers, AI-driven analytics, and network protection mechanisms.
11. Sustainable Cities and Communities	By enhancing home security and safety, the system contributes to safer smart homes and communities, reducing crime risks and ensuring proactive security monitoring.
16. Peace, Justice and Strong Institutions	The honeypot-based intrusion detection system improves cybersecurity, helping to mitigate cyber threats and ensuring secure digital environments.

SDG 3

SDG 9

SDG 11

SDG 16